

EFFICIENT IMPLEMENTATION OF THE HYBRID METHOD FOR STOCHASTIC SIMULATION OF BIOCHEMICAL SYSTEMS

SHUO WANG, YANG PU, LAYNE WATSON, AND YANG CAO

1. INTRODUCTION

Living cells are regulated by complicated biological networks. To study the mechanism of these networks, systems biologists often use chemically reacting systems and ordinary differential equations (ODEs) to model the behavior of cellular networks. In living cells, however, reacting molecules participate in chemical reactions in a discrete and stochastic manner rather than a continuous and deterministic one. This stochastic feature results in noise, which sometimes has significant impacts on living cells. In order to take the noise into consideration, stochastic simulation methods are needed. Among them the most famous one is Gillespie’s stochastic simulation algorithm (SSA) [3, 4] for well-stirred chemically reacting systems. However, as models become more complex, the SSA often becomes computationally expensive because it simulates every reaction firing. In biochemical systems, some reactions involve species of large populations or have large reaction rates. These reactions tend to dominate the CPU time in an SSA simulation. In order to speed up the SSA, several approximation strategies have been proposed. One group of approximation methods tries to take advantage of the multiscale property among reactions: Some reactions are much faster than others and these fast reactions may reach partial equilibrium [2], or quasi steady

2010 *Mathematics Subject Classification.* Primary 65L06; Secondary 92C40.

Key words and phrases. Hybrid Method, Stochastic Simulation Algorithm, Ordinary Differential Equation Solver.

state [14] during each slow reaction firing. These methods include the quasi steady state SSA method [14] and the slow-scale SSA method [2]. Another group of approximation algorithms was proposed to take advantage of the multiscale characteristics among species populations. The tau-leaping methods [5] belong to this group. To combine the tau-leaping method and the SSA, Cao et al. [1] proposed a hybrid method to partition species into two sets according to their population numbers. Reactions involving species whose population are less than a threshold are simulated by the SSA and other reactions are simulated by the tau-leaping method.

Of course, multiscale features among reactions and species populations are connected. Often fast reactions are related to species with large populations. Based on this observation, Haseltine and Rawlings [8] developed a strategy to partition a system into a slow reaction group and a fast reaction group. The fast reaction group consists of fast reactions whose reactants are of large populations and are modeled by ODEs or chemical Langevin equations (CLEs). The rest reactions, either the reaction is slow or one of the involved species is in a small population, are put in the slow reaction group and are modeled and simulated by the SSA. Recently a new partitioning strategy was proposed to put all fast reactions, no matter its reactants are in large or small populations, into the fast reaction group and simulated by ODEs. This strategy made Haseltine and Rawling’s hybrid method more efficient and practical when applied to biochemical systems. On the other hand, the implementation details for the hybrid method were not discussed in literature at all although it is critical for the success of the hybrid method.

To implement Haseltine and Rawling’s hybrid method, numerical solvers for stiff ODEs are always needed due to the multiscale features. There are two major groups of ODE

solvers for stiff problems: one step methods and multistep methods. The former group include the famous Runge-Kutta methods, while the latter group include Adams methods and BDF methods, which many popular stiff ODE solvers are based on. ODE solvers based on multistep methods have received more attention in literature since these methods usually lead to smaller linear system size than the one step methods and thus are considered faster than one step methods. However, when applying the hybrid method to biochemical systems, there is a very special feature. ODE integration is often interrupted by discrete slow reactions. In that case, ODE solvers based on multistep methods have to reduce orders and stepsizes to handle the discontinuity resulted from discrete slow reactions. When slow reactions fire frequently, it is a difficult job for multistep methods to maintain its high order and large stepsizes. Then the efficiency of multistep methods will be heavily affected. On the contrary, step sizes of one step methods will not be affected by the discontinuity resulted from discrete slow reactions. They are usually greater than stepsizes selected by multistep methods. Therefore, we conjecture that one step methods should be a better choice when the hybrid method is concerned. But before we draw a solid conclusion, we need to study what is the most efficient way to implement the hybrid method using one step ODE solvers, and we need to perform numerical experiments on typical biological models to verify/reject our conjecture.

In this paper, we present our study of the implementation details related to the hybrid method. We first developed an efficient strategy for the root finding function in any ODE solver. This strategy utilizes a special feature in the hybrid method: the event function is monotonously increasing. With this strategy, the event handling will be more efficient and

can be applied to both one step methods and multistep methods. To investigate the accuracy and efficiency of hybrid method in terms of different ODE solvers, we did numerical experiments based on three biological models. The ODE solvers we tested are RADAU5 [7], DASSL [13], and DLSODAR [10], where RADAU5 is an implicit one step Runge-Kutta method ODE solver, DASSL and DLSODAR are ODE solvers based on multistep methods. Numerical results demonstrate that the hybrid method with these three ODE solves have similar accuracy but different efficiency performance. For the two multistep ODE solvers, DLSODAR has better performance. In the first two experiments, RADAU5 shows its advantage when the integration process is interrupted frequently by slow reactions. In the third experiments, RADAU5 does not perform as well as expected when slow reaction firing frequency increases. Performance data collected in numerical experiments indicate that the drastically growing CPU time on Jacobian matrix calculation largely weakens RADAU5's efficiency.

This paper is organized as follows. In Section 2, we briefly review Gillespie's SSA and introduce the hybrid method and ODE solvers applied in our numerical experiments. In Section 3, we propose an efficient event handling function in terms of inverse interpolation. The accuracy of this method is also discussed. Three numerical experiments and corresponding results are shown in Section 4. In Section 5, we give our conclusion and further discussion.

2. BACKGROUND

2.1. Stochastic simulation algorithm. We consider a well-stirred system of N chemical species $\{S_1, \dots, S_N\}$ and M chemical reactions $\{R_1, \dots, R_M\}$. We define the current state of system as a vector $\mathbf{x} = (x_1, \dots, x_N)$, where x_i is the number of S_i molecules. Each

reaction channel R_j is characterized by its propensity function $a_j(\mathbf{x})$ and its state-change vector $\mathbf{v}_j = (v_{1j}, \dots, v_{Nj})$. The probability of reaction R_j firing in the next infinitesimal time dt is given as $a_j(\mathbf{x})dt$, and v_{ij} S_i molecules will be changed by one R_j reaction. The procedure of SSA is presented as follows.

- 1) In state \mathbf{x} at time t , compute propensity values $a_j((x))$, and the sum $a_0(\mathbf{x}) \equiv \sum_{j=1}^M a_j(\mathbf{x})$.
- 2) Generate a time increment τ as a sample of the exponential random variable with mean $1/a_0(\mathbf{x})$.
- 3) Generate a reaction index j based on the probability function $a_j(\mathbf{x})/a_0(\mathbf{x})$ ($j = 1, \dots, M$).
- 4) Update time $t \leftarrow t + \tau$ and state $\mathbf{x} \leftarrow \mathbf{x} + \mathbf{v}_j$.
- 5) Return to step 1) if stopping condition is not reached.

Although SSA is mathematically exact, the time cost by SSA may become very expensive when the size of a biochemical system increases, since SSA should simulate every reaction event and the time increment τ may be very small corresponding to a large $a_0(\mathbf{x})$. Thus it is necessary to develop more efficient simulation methods.

2.2. Hybrid method. The hybrid method follows the strategy proposed by Haseltine and Rawlings. Given a system of N species $\{S_1, \dots, S_N\}$ and M reactions $\{R_1, \dots, R_M\}$, these M reactions are partitioned into two subsets. Reactions in subset S_{fast} are formulated by ODEs, and reactions in subset S_{slow} are simulated with SSA. Let $a_i(\mathbf{x}, t)$ be the propensity function of i -th reaction in S_{slow} , where \mathbf{x} is the state at time t , and its state-change vector

$\mathbf{v}_i = \{v_{i1}, \dots, v_{iN}\}$. Let τ be the jump interval of the next stochastic reaction, and μ be its reaction index. Set $t = 0$. The hybrid method simulate the system as follows:

- 1) Generate two uniform random numbers r_1 and r_2 in $U(0, 1)$.
- 2) Integrate the ODE system and the integral equation:

$$(1) \quad \int_t^{t+\tau} a_{tot}(\mathbf{x}, s) ds + \log(r_1) = 0,$$

where $a_{tot}(\mathbf{x}, t)$ is the sum of propensities of reactions in S_{slow} .

- 3) Determine μ as the smallest integer satisfying

$$(2) \quad \sum_{i=1}^{\mu} a_i(\mathbf{x}, t) > r_2 a_{tot}(\mathbf{x}, t).$$

- 4) Update $\mathbf{x} \leftarrow \mathbf{x} + \mathbf{v}_\mu$.
- 5) Return to step 1) if stopping condition is not reached.

In step 2), different from Haseltine and Rawlings's strategy of adding a propensity of "no reaction", we choose a more direct strategy to get the jump interval τ . Suppose that the ODE system is given by

$$(3) \quad \mathbf{x}' = f(\mathbf{x}).$$

We simply add an integration variable z and an equation

$$(4) \quad z' = a_{tot}(\mathbf{x}), z(t) = \log(r_1),$$

where we note that $\log(r_1)$ is negative. During the simulation, each step starts at time t and numerically integrates ODEs (3) and (4). When $z(t + \tau) = 0$, the ODE integration stops, where τ is the solution to (1). This integration process can be conveniently handled by standard ODE solvers combined with root-finding functions. Note that since z is an integration variable, one may choose to omit it from the error control mechanism [13]. Adding this extra variable will not greatly affect the efficiency of ODE solution.

2.3. ODE solvers. When applying the hybrid method to simulate biochemical systems, all reactions in the subset S_{fast} and the equation (4) need to be solved by an ODE solver. Because of the multiscale property of biochemical systems, ODE solvers for stiff problems are usually preferred. There are two types of widely-used ODE solvers (in Fortran) for stiff problems, one based on one step methods and the other based on multistep methods.

For one step methods, in each step to integrate from t_n to t_{n+1} , the integration only uses the value $\mathbf{x}(t_n)$. For stiff systems, implicit methods are usually used. RADAU IIA (RADAU5) is an implicit Runge-Kutta method of stage three and order five for stiff ODEs, which was originally proposed by Axelsson in 1969 [7]. In each step, RADAU5 uses the current state at time t to calculate the state at time $t + h$, where h is the step size. The Fortran code we used for RADAU5 was written by E. Hairer and G. Wanner [7]. It offers efficient step size control strategy and continuous output, but no root finding function. However, this work can be done with the assistant of function CONTR5 when calling function SOLOUT. More details can be found in the user guide of the RADAU5 code.

DASSL and DLSODAR are widely-used multistep ODE solvers for stiff problems. In contrast to one step methods, multistep methods use state values at several previous time points

to calculate the state value at t_{n+1} . Adams method and BDF (backward differentiation formulas) method are two popular multistep methods. The idea of Adams method is to approximate $\mathbf{x}(t_{n+1})$ using derivatives from several previous time points, while the BDF method is based on an interpolation polynomial using state values at previous time points to approximate the derivative at t_{n+1} . DASSL was developed by Linda R. Petzold [13]. It follows the idea of BDF methods. Similar as RADAU5, DASSL also does not directly support root handling function. DLSODAR is an ODE solver developed by Alan C. Hindmarsh and Linda R. Petzold [10]. A great feature of DLSODAR is that it can automatically switch between Adams methods and BDF methods based on characteristics of underlying ODEs. This property allows DLSODAR to have a better performance compared with DASSL [12]. It also offers root-finding function. Thus it is more convenient for event handling.

3. INVERSE INTERPOLATION

In the hybrid method, the event that a slow reaction in S_{slow} fires is important and has significant impact on the accuracy of the simulations. Assume the integration for the equation (4) starts at time t . For a slow reaction to fire at t_{event} , we should have $z(t_{event}) = 0$. In order to obtain the solution t_{event} , ODE solvers have to equip with root-finding functions. DLSODAR has a root-finder based on Illinois Algorithm [9], a variant of secant method, which returns the leftmost solution in terms of iterations. But for RADAU5 and DASSL, root finding functions are not directly provided. In order to compare the performance of RADAU5, DASSL, and DLSODAR for the hybrid method, we need to develop an efficient root finding mechanism for them. Generally, a root finding function in an ODE solver needs to develop an interpolation polynomial using state values at time points (for multistep methods) or stage

points (for Runge-Kutta methods). However, root finding function in the hybrid method has a very special feature, which makes the inverse interpolation method a very good choice. In equation (4), the derivative of $z(t)$ is given by $a_{tot}(\mathbf{x})$, the sum of propensities of slow reactions. Thus $z'(t)$ is always positive. It implies that $z(t)$ is a monotonous function with respect to time t . So there must exist an inverse function for $z(t)$. We denote the inverse by $t = T(z)$. Thus t_{event} , the solution of $z(t) = 0$, can be given by $t_{event} = T(0)$. In the following we give the details on how the inverse interpolation method is implemented for RADAU5 and DASSL.

3.1. Inverse interpolation in RADAU5. RADAU AII is a three-stage five-order implicit Runge-Kutta method. For an ordinary differential equation

$$(5) \quad y'(t) = f(t, y),$$

the value at time $t + h$ is computed based on the calculated value at time t in the previous step and corresponding formulas are as follows

$$(6) \quad \begin{aligned} g_i &= y(t) + h \sum_{j=1}^3 a_{ij} f(t + c_j h, g_j), \quad i = 1, 2, 3, \\ y(t + h) &= y(t) + h \sum_{j=1}^3 b_j f(t + c_j h, g_j), \end{aligned}$$

where h is the step size, a_{ij} , b_j and c_j are coefficients in RADAU AII [7]. For $i = 1, 2$, g_i is an approximation of $y(t + c_j h)$. For $i = 3$, a_{3j} shares the same value with b_j and c_3 is equal to 1, therefore g_3 gives the value of y at time $t + h$. All g_i 's are calculated using implicit method.

When running the hybrid method with RADAU5, the integration process showed in equation (6) is applied to both fast reactions described as ODEs and the equation (4) step by step when z is less than 0. If $z \geq 0$ is detected at time $t + h$, we call a root-finding function using inverse interpolation to find the time $t < t_{event} \leq t + h$ by solving $z(t_{event}) = 0$. For example, the inverse interpolation function $T_1(z)$ can be constructed in the Lagrange form as

$$\begin{aligned}
(7) \quad T_1(z) = & \frac{(z-g_1)(z-g_2)(z-g_3)}{(z_t-g_1)(z_t-g_2)(z_t-g_3)}t \\
& + \frac{(z-z_t)(z-g_2)(z-g_3)}{(g_1-z_t)(g_1-g_2)(g_1-g_3)}(t + c_1h) \\
& + \frac{(z-z_t)(z-g_1)(z-g_3)}{(g_2-z_t)(g_2-g_1)(g_2-g_3)}(t + c_2h) \\
& + \frac{(z-z_t)(z-g_1)(z-g_2)}{(g_3-z_t)(g_3-g_1)(g_3-g_2)}(t + h)
\end{aligned}$$

The solution to $z(t_{event}) = 0$ can be obtained as $t_{event} = T_1(0)$.

3.2. Inverse interpolation in DASSL. DASSL is an ODE solve based on the BDF method. The solution to equation (5) at step n can be obtained as

$$(8) \quad y_n = \sum_{i=1}^q \alpha_i y_{n-i} + h\beta_0 f(t_n, y_n)$$

where α_i and β_0 are coefficients, h is the step size, and q is the order indicating how many previous time points are needed. In DASSL, the order q varies from one to five.

Different from the one step method used in RADAU5, DASSL does not calculate the value between y_{n-1} and y_n . Therefore we have to adopt a little different inverse interpolation strategy in DASSL. During the process of integration, we use two vectors of length six to record the values of z for equation (4) and their corresponding time t . When $z_n > 0$ is detected, we apply z_n and previous q results z_{n-1}, \dots, z_{n-q} as independent variables and

t_n, \dots, t_{n-q} as dependent variables to construct a polynomial $T_2(z)$ with highest order q .

The solution t_{event} where $z(t_{event}) = 0$ is got at $t_{event} = T_2(0)$.

3.3. Accuracy of inverse interpolation. In this part, we will briefly discuss the accuracy of inverse interpolation in both RADAU5 and DASSL.

First, we give an accuracy analysis of inverse interpolation for RADAU5. Let $z(t)$ be the exact value of equation (4) at time t and z_t is the numerical solution to $z(t)$. According to Lemma 7.5 in [6], we have

$$\begin{aligned}
 z_t &= z(t) + O(h^5), \\
 g_1 &= z(t + c_1 h) + O(h^3), \\
 g_2 &= z(t + c_2 h) + O(h^3), \\
 g_3 &= z(t + h) + O(h^5).
 \end{aligned}
 \tag{9}$$

In order to simplify this analysis, we assume that $z_t < g_1 < g_2 < g_3$. Let t^* be the exact time increment where $z(t^*) = 0$ and \hat{t} be the approximation obtained from equation (7). If we replace z_t and g_i by $z(t)$ and $z(t + c_i h)$ in $T_1(0)$ respectively, we have an approximation denoted by t_z . The error can be represented as

$$err(\hat{t}) = (\hat{t} - t_z) + (t_z - t^*)
 \tag{10}$$

This error is composed of two parts. The first part on the right hand side gives the error from approximating $z(t)$ and $z(t + c_i h)$ by z_t and g_i . The second part is the interpolation error. To obtain the error bound for the first part, we use the Lagrange form (7) of $T_1(z)$ and let $z = 0$. Let z_0 denote $z(t)$ and z_i denote $z(t + c_i h)$, by ignoring the higher order of h ,

we have

$$\begin{aligned}
(11) \quad \frac{g_1 g_2 g_3}{(z_t - g_1)(z_t - g_2)(z_t - g_3)} t &= \frac{(z_1 + O(h^3))(z_2 + O(h^3))(z_3 + O(h^5))}{3} t \\
&\prod_{i=1}^3 (z_0 - z_i + O(h^3)) \\
&= \frac{z_1 z_2 z_3}{(z_0 - z_1)(z_0 - z_2)(z_0 - z_3)} t + O(h^3)
\end{aligned}$$

For the rest two items in $T_1(0)$, we have similar results as in (11). Therefore, $\hat{t} - t_z = O(h^3)$.

Moreover, the inverse interpolation based on $z(t + c_i h)$ has an error $O(h^4)$ that indicates $t_z - t^* = O(h^4)$. We can get the total error $err(\hat{t}) = O(h^3)$.

We can apply a similar accuracy analysis to the inverse interpolation in DASSL. We define $z(t_i)$ as the exact solution to equation (4) at time t_i and z_i as the numerical solution calculated from equation (8) using q_i previous solutions. Let t^* be the absolute time where $z(t^*) = 0$, \hat{t} be the approximation given by $T_2(0)$ and t_z be the solution after replacing z_i by $z(t_i)$ in T_2 . Theory 2.4 in [6] gives that $z_i = z(t_i) + O(h^{q_i})$. We thus have $\hat{t} - t_z = O(h^{\min(q_i)})$. On the other hand, we assume that $q + 1$ exact values are used to construct inverse interpolation polynomial. We have $t_z - t^* = O(h^q)$. Therefore the order of solution \hat{t} is $\min(q_i, q)$. In DASSL, q varies from one to five. The inverse interpolation in DASSL has a maximum order five.

4. NUMERICAL EXAMPLES

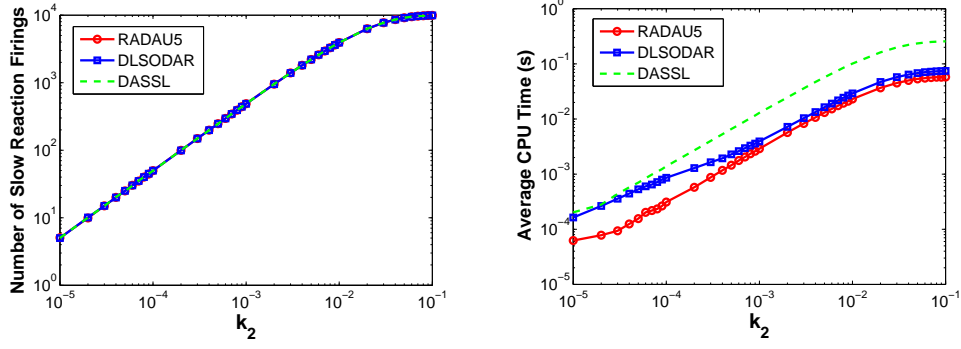
In a hybrid method simulation, each time when there is a slow reaction firing, the ODE solver should stop and restart after the state is updated. For a multistep ODE solver like DASSL, the restart procedure may have great overhead if slow reactions fire too often because it has to reduce to a first order method with a small stepsize. This process has a

heavy impact on the efficiency. For a Runge-Kutta solver like RADAU5, the impact should be small. In this section we compare the accuracy and efficiency of hybrid methods with three different ODE solvers: RADAU5, DASSL, and DLSODAR. Numerical results for three examples are present. One example is a simple model involving three species and three reactions. The second example is a gene regulation process, and the third one is a cell cycle model presented in Liu et al. [11]. Because they are stochastic simulations, the number of species and simulation CPU time vary in each run. We therefore recorded the number of slow reactions firing times and CPU time for each simulation. Accuracy and efficiency are measured in terms of average results, respectively. We set absolute error tolerance as 10^{-6} and relative error tolerance as 10^{-3} for all three ODE solvers. The simulations were performed on a 2.30GHz Intel Core Linux workstation.

4.1. **Example one.** We first consider a simple system



where $k_1 = 0.5$, $k_{-1} = 1.5$ and k_2 varies from 10^{-5} to 10^{-1} . The initial values of three species are $x_1(0) = 7500$, $x_2(0) = 2500$, and $x_3(0) = 0$. Here the first two reversible reactions are considered as fast reactions and simulated by ODE solvers, and the last reaction is a slow reaction and handled by SSA. This system were simulated 1,000 times with $T = 200$ as the final time using ODE solvers RADAU5, DASSL, and DLSODAR, respectively. The average CPU time and the number of firing times for the third reaction are shown in Figure 1.



(a) Average firing times of the third reaction on example one

(b) Average CPU time of hybrid method with three ODE solvers on example one

FIGURE 1. Accuracy and efficiency comparison of hybrid method with three ODE solvers on example one

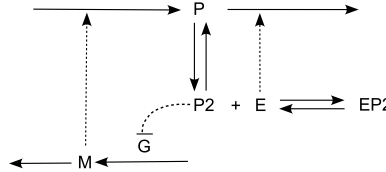


FIGURE 2. Diagram for gene regulation model

Figure 1(a) shows the mean value of the third reaction firing number and Figure 1(b) gives the mean value of CPU times, corresponding to different k_2 values. In Figure 1(a), the three curves match with each other very well. It suggests that given the same error tolerance, hybrid methods with RADAU5, DASSL and DLSODAR have similar accuracy for this model. But from Figure 1(b), it is easy to see that the average running times of the hybrid method with the three ODE solvers are different. RADAU5 performs the best for this example, while DASSL is the worst. For DLSODAR, when $k_2 < 10^{-4}$ the curve is close to the one for DASSL and when $k_2 > 10^{-3}$ the curve is close to the one for RADAU5. But there is no crossover. So for this simple system, RADAU5 is the best choice for the hybrid method.

4.2. Gene regulation model. The second model is a gene regulation model, which was first given in the appendix of paper [11]. The diagram for this model is shown in Figure 2.

TABLE 1. Reactions in the Gene Regulation Model

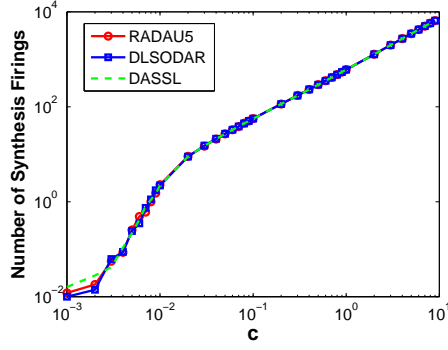
| Reaction | Rate Constant | Partition |
|--------------------------|----------------|-----------|
| $g \rightarrow g + m$ | $k_1 = 0.5$ | SSA |
| $m \rightarrow m + p$ | $k_2 = 60$ | ODE |
| $p \rightarrow \phi$ | $k_3 = 0.05$ | ODE |
| $m \rightarrow \phi$ | $k_4 = 0.05$ | SSA |
| $p + p \rightarrow p2$ | $k_5 = 0.001$ | ODE |
| $p2 \rightarrow p + p$ | $k_6 = 40$ | ODE |
| $p2 + g \rightarrow gi$ | $k_7 = 40$ | ODE |
| $gi \rightarrow p2 + g$ | $k_8 = 1000$ | ODE |
| $E + p2 \rightarrow Ep2$ | $k_9 = 0.4$ | ODE |
| $Ep2 \rightarrow E + p2$ | $k_{10} = 0.5$ | ODE |
| $E + p \rightarrow Ep$ | $k_{11} = 0.2$ | ODE |
| $Ep \rightarrow E + p$ | $k_{12} = 10$ | ODE |
| $Ep \rightarrow E$ | $k_{13} = 10$ | ODE |

TABLE 2. Initial Value of Reactants in the Gene Regulation Model

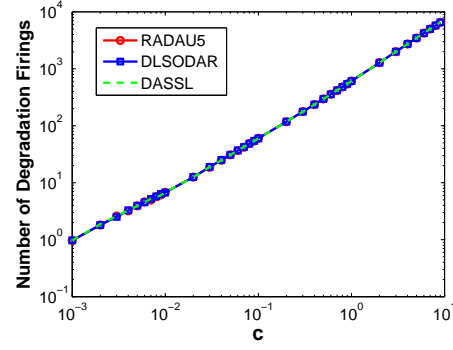
| Reactant | g | gi | m | p | p2 | E | Ep | Ep2 |
|---------------|---|----|----|------|-----|-----|----|-----|
| Initial Value | 1 | 0 | 10 | 1000 | 100 | 100 | 0 | 0 |

In this model, the synthesis of protein p is controlled by the expression of gene g . Protein p also can form homodimer $p2$, which can bind to the promotor site of its own gene and repress the expression (a negative feedback). In addition, $p2$ also binds and inhibits the enzyme E , which catalyzes the degradation of protein p (a positive feedback). Oscillation of total level of protein ($p_{total} = p + 2 * p2$) occurs during the simulation. Reactions, rate constants and partition are given in Table 1. The initial value of species are shown in Table 2.

In this model, synthesis and degradation of the mRNA m are simulated by the SSA. To test the influence of slow reaction firing frequency on the performance of hybrid methods with different ODE solvers, we introduce a scale factor c and multiply it to k_1 and k_4 . We let c vary from 10^{-3} to 10. The larger c is, the more frequently an ODE solver has to restart. With $T = 2,000$ as the final time, this system were simulated 1,000 times using the hybrid method with ODE solvers RADAU5, DASSL, and DLSODAR, respectively. The



(a) Average firing times of m synthesis reaction on gene regulation model



(b) Average firing times of m degradation reaction on gene regulation model

FIGURE 3. Accuracy comparison of hybrid method based on three ODE solvers on gene regulation model

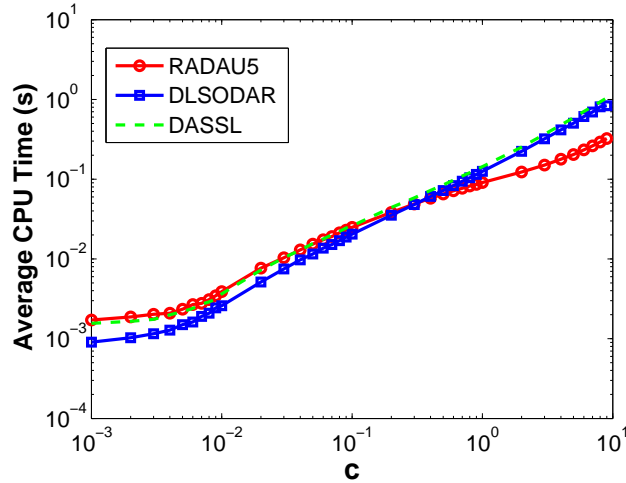


FIGURE 4. Efficiency comparison of hybrid method based on three ODE solvers on gene regulation model

mean firing numbers of synthesis and degradation and average CPU time are shown in Figure 3 and Figure 4.

From Figure 3(a), we observe that when c is less than 10^{-2} , the system's stochastic property has an obvious impact on the synthesis of m . As c increases, the curves become smooth and match with each other very well. Also in Figure 3(b), we can see that hybrid methods with the three ODE solvers give similar results. Same as the result shown in Figure 1(b), in Figure

TABLE 3. ODE system of hybrid cell cycle model

$$\begin{aligned}
\frac{d}{dt}V &= \mu V \\
\frac{d}{dt}\langle X \rangle &= k_{sx}M_xV - k_{dx}\langle X \rangle - \frac{k_{dxy}\langle X \rangle \langle Y \rangle}{V} \\
\frac{d}{dt}\langle Y_T \rangle &= k_{sy}M_yV - k_{dy}\langle Y_T \rangle \\
\frac{d}{dt}\langle Y \rangle &= k_{sy}M_yV - k_{dy}\langle Y \rangle + \frac{(k_{hy}V + k_{hyz}\langle Z \rangle)(\langle Y_T \rangle - \langle Y \rangle)}{J_{hy}V + \langle Y_T \rangle - \langle Y \rangle} \\
&\quad - \frac{k_{pyx}\langle X \rangle \langle Y \rangle}{J_{pyx}V + \langle Y \rangle} \\
\frac{d}{dt}\langle Z \rangle &= k_{sz}M_zV - k_{dz}\langle Z \rangle
\end{aligned}$$

TABLE 4. SSA system of hybrid cell cycle model

| Reaction | Propensity Function |
|------------------------|---|
| $\phi \rightarrow M_x$ | $k_{smx}V$ |
| $M_x \rightarrow \phi$ | $k_{dmx}M_x$ |
| $\phi \rightarrow M_y$ | k_{smy} |
| $M_y \rightarrow \phi$ | $k_{dmy}M_y$ |
| $\phi \rightarrow M_z$ | $k_{smz} + \frac{k_{smzx}\langle X \rangle^2}{(J_{smzx}V)^2 + \langle X \rangle^2}$ |
| $M_z \rightarrow \phi$ | $k_{dmz}M_z$ |

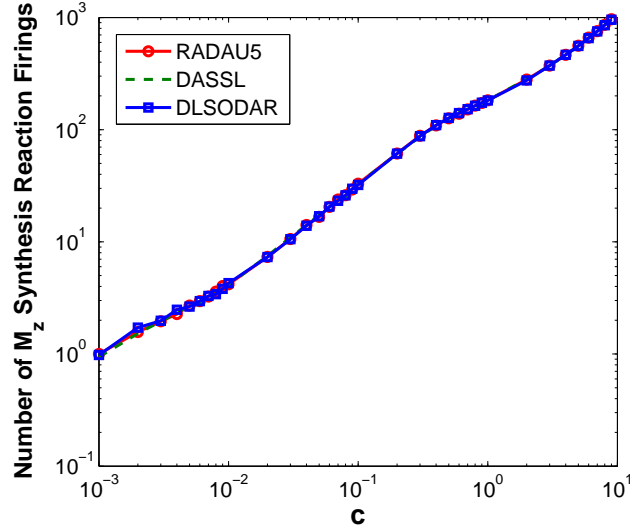
Let X denote Cdk1-Clb complexes, Y denote Cdh1 and Z denote the lumped reaction of Cdc20 and Cdc14. In this model, X inactivates Y by phosphorylating it, and unphosphorylated Y catalyzes X 's degradation. X also promotes the synthesis of Z and Z catalyzes the dephosphorylation of phosphorylated Y . Therefor, X and Y are involved in a mutual antagonism, which creates a bistable switch.

Based on this three-variable cell cycle model, Liu et al. [11] built a hybrid stochastic cell cycle model by introducing six stochastic reactions for the synthesis and degradation of three mRNA variables. In this hybrid model, for the ODE part, besides X , Y , and Z , the variable V denotes the volume of cell, and Y_T is the total amount of Y (unphosphorylated and phosphorylated). For the SSA part, M_x , M_y , and M_z are three mRNA variables for X , Y , and Z , respectively. Details of the model are listed in Table 3, Table 4 and Table 5.

To test the influence of slow reactions on the performance of the hybrid method with different ODE solvers, we introduce a scale factor c varying from 10^{-3} to 10, and multiply it to the

TABLE 5. Parameter values of hybrid cell cycle model

| Parameter | Value (min^{-1}) | Parameter | Value ($fL^{-1} \cdot \text{min}^{-1}$) |
|------------|-----------------------------|------------|---|
| μ | 0.006 | k_{sx} | 1.53 |
| k_{dx} | 0.04 | k_{sy} | 1.35 |
| k_{dy} | 0.02 | k_{hy} | 29.7 |
| k_{hyz} | 7.5 | k_{sz} | 1.35 |
| k_{pyx} | 1.88 | k_{smx} | 1.04 |
| k_{dz} | 0.1 | Parameter | Value ($fL \cdot \text{min}^{-1}$) |
| k_{dmx} | 3.5 | k_{dxy} | 0.00741 |
| k_{smy} | 7.0 | Parameter | Value (fL^{-1}) |
| k_{dmy} | 3.5 | J_{hy} | 5.4 |
| k_{smz} | 0.001 | J_{pyx} | 5.4 |
| k_{smzx} | 10.0 | J_{smzx} | 756 |
| k_{dmz} | 0.15 | | |

FIGURE 6. Average firing times of M_z synthesis reaction of hybrid method with three ODE solvers on cell cycle model

propensity functions of the six stochastic reactions in the SSA subsystem. With $T = 2000$ as the final time, for each set of parameter the system was simulated 1,000 times using hybrid methods with RADAU5, DASSL, and DLSODAR, respectively. The mean firing numbers of synthesis and degradation reactions and average CPU time are shown in Figure 6 and Figure 7.

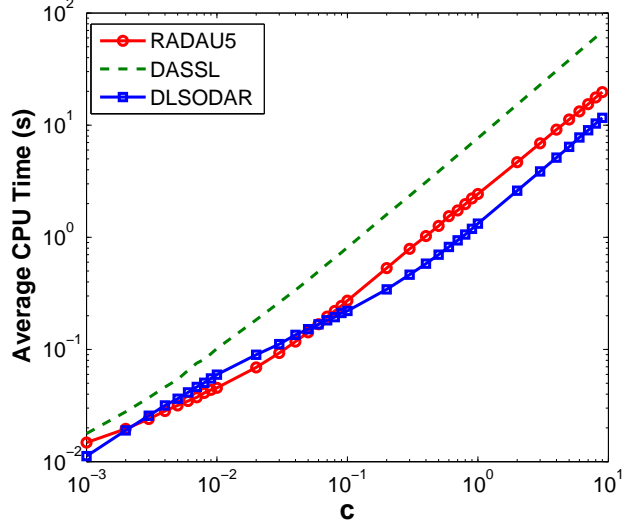


FIGURE 7. Average CPU times of hybrid method with three ODE solvers on cell cycle model

For the comparison of accuracy, we just plot the results for M_z synthesis reaction in Figure 6, the propensity function of which contains a hill function. Similar as the previous two examples, hybrid methods with three ODE solvers give similar accuracy results. Figure 7 shows the average CPU times of hybrid methods with three ODE solvers. DASSL performs the worst compared with the other two ODE solvers. The green dash curve is on the top as c increases. If we focus on the red dot curve (RADAU5) and the blue square curve (DLSODAR), we note that these two curves have two crossovers, different from the no crossover pattern in Figure 1(b) and the one crossover pattern in Figure 4. When $0.002 < c < 0.06$, hybrid method with RADAU5 takes less time. But on other scales, hybrid method with DLSODAR has higher efficiency.

5. CONCLUSION AND DISCUSSION

The hybrid method which combines both continuous model and stochastic model has received more and more attentions from biochemical system simulation community. It not only

keeps the stochastic property of biochemical system but also costs less time to accomplish simulations by dealing with the continuous part with more efficient approaches. When applying the hybrid method, the continuous model is usually handled by ODE solvers. On the fortran platform, DLSODAR, an ODE solver based on multistep method is widely used. But when integration is interrupted frequently by the stochastic subsystem, the efficiency of multistep ODE solvers may drop heavily, since the frequent restarting process inhibits ODE solvers to use higher orders, while the one step method ODE solver RADAU5 may have its advantages under this situation.

In this paper, we developed a root finding strategy, which is particularly designed for the hybrid method, and applied them to RADAU5 and DASSL. With the modification, we were able to conduct three groups of numerical experiments to compare the accuracy and efficiency of hybrid methods with three ODE solvers. The results of the first two experiments demonstrate the advantage of RADAU5 when systems contain frequently firing slow reactions. However, in the third experiment on a cell cycle model, DLSODAR has better performances with respect to low and high firing frequency of slow reactions. The reason why RADAU5 no longer holds its advantages for the cell cycle model is intriguing to explore.

RADAU5 is an ODE solver based on the implicit Runge-Kutta method. In order to calculate the state values at the next step, Newton's iteration, the main part of RADAU5, requires accurate Jacobian matrix. Therefore, beside Newton's iteration, the calculation of Jacobian matrix is also an important part, which takes up a large percent of CPU time. Each time when RADAU5 starts, the Jacobian matrix should be calculated immediately. During the integration, if errors are accepted, there is no need to recalculate Jacobian matrix. For the first two models, each set of ODEs involves no more than two species. To obtain the Jacobian

matrix is not a hard work. But it is not the same for the cell cycle model. In the cell cycle model, the highly nonlinear terms make the calculation more expensive. With the aid of collected information from running data, we observed that as the restart number grows, RADAU5 spends more and more time on Jacobian matrix, rising from forty percent to sixty percent. However, the multistep method DLSODAR avoids such a process. DLSODAR has a method selection strategy, which automatically switches between Adams methods and BDF methods. When parameter c is beyond a threshold, both RADAU5 and DLSODAR will stop in each step. In such condition, DLSODAR always chooses Adams method to enhance the efficiency. No Jacobian matrix calculation is needed any more. This property largely reduces computation complexity. Therefore, although the average step size of RADAU5 is larger than DLSODAR, the benefit is gradually counteracted by the burden of calculation of Jacobian matrix. That is why the curve referring average CPU time of hybrid method with RADAU5 intercepts the one of hybrid method with DLSODAR twice. When making use of hybrid methods, we should carefully partition systems. If the sum of propensities of slow reaction is large enough, which forces ODE solvers to stop very frequently, DLSODAR may be the first to be considered.

To make the analysis easier in section 3, we made an assumption that $t_z < g_1 < g_2 < g_3$, since g_i is an approximation to $z(t + c_i h)$ which is an monotonous function. Each time when ODE solvers are interrupted at time $t + h$, we can guarantee $t_z < 0$ and $g_3 > 0$. Of course, the time t_{event} when slow reaction fires must be within time interval $[t, t + h]$. However, in practice, we are not sure what will be the situations for g_1 and g_2 . If values of these four variables have the relationship as we assumed, the inverse interpolation will work very well. If this assumption is not satisfied, inverse interpolation could be dangerous. Under this condition,

these four variables no longer describe a monotonous function. The result may be far from the true value, for instance we may get $t_{event} > t + h$ or $t_{event} < t$, which is an obvious wrong output. So far we have not observed such big errors appearing in our numerical experiments. But, in order to prevent this bad inverse interpolation from affecting the accuracy heavily, more work needs to be done to improve the root finding mechanism.

Acknowledgements This work was partially supported by the National Science Foundation under awards DMS-1225160, CCF-0953590, and CCF-1526666 and the National Institutes of Health under award GM078989.

REFERENCES

1. Yang Cao, Daniel T. Gillespie, and Linda R. Petzold, *Avoiding negative populations in explicit poisson tau-leaping*, Journal of Chemical Physics **123** (2005), no. 5, 054104.
2. ———, *The slow-scale stochastic simulation algorithm*, Journal of Chemical Physics **122** (2005), no. 1.
3. Daniel T. Gillespie, *A general method for numerically simulating the stochastic time evolution of coupled chemical reactions*, Journal of Computational Physics **22** (1976), no. 4, 403–434.
4. ———, *Exact stochastic simulation of coupled chemical reactions*, Journal of Physical Chemistry **81** (1977), no. 25, 2340–2361.
5. ———, *Approximate accelerated stochastic simulation of chemically reacting systems*, Journal of Chemical Physics **115** (2001), no. 4.
6. Ernst Hairer, Syvert P. Nørsett, and Gerhard Wanner, *Solving ordinary differential equation i, nonstiff problems*, Springer, 2011.
7. Ernst Hairer and Gerhard Wanner, *Solving ordinary differential equation ii, stiff and differential-algebraic problems*, Springer, 2004.
8. Eric L. Haseltine and James B. Rawlings, *Approximate simulation of coupled fast and slow reactions for stochastic chemical kinetics*, Journal of Chemical Physics **117** (2002), no. 15, 6959–6969.

9. Kathie L. Hiebert and Lawrence F. Shampine, *Implicitly defined output points for solutions of odes*, Tech. report, Sandia Laboratories, 1980.
10. Alan C. Hindmarsh, *Odepack, a systematized collection of ode solvers*, IMACS Transactions on Scientific Computing, vol. 1, September 1983, pp. 55–64.
11. Zhen Liu, Yang Pu, Fei Li, Clifford A. Shaffer, Stefan Hoops, John J. Tyson, and Yang Cao, *Hybrid modeling and simulation of stochastic effects on progression through the eukaryotic cell cycle*, Journal of Chemical Physics **136** (2012), no. 3.
12. Linda Petzold, *Automatic selection of methods for solving stiff and nonstiff systems of ordinary differential equations*, SIAM Journal on Scientific and Statistical Computing **4** (1983), no. 1, 136–148.
13. Linda R. Petzold, *A description of dassl: A differential/algebraic system solver*, The IMACS World Congress, vol. 1, September 1982, p. 65.
14. Christopher V. Rao and Adam P. Arkin, *Stochastic chemical kinetics and the quasi steady-state assumption: application to the gillespie algorithm*, Journal of Chemical Physics **118** (2003), no. 11.
15. John J. Tyson and Bela Novak, *Temporal organization of the cell cycle*, Current Biology **18** (2008), no. 17, R759–R768.

ABSTRACT. Stochastic effect in cellular systems has been an important topic in systems biology. Stochastic modeling and simulation methods are important tools to study stochastic effect. Given the low efficiency of stochastic simulation algorithms, the hybrid method, which combines an ordinary differential equation (ODE) system with a stochastic chemically reacting system, shows its unique advantages in the modeling and simulation of biochemical systems. The efficiency of hybrid method is usually limited by reactions in the stochastic subsystem, which are modeled and simulated using Gillespie's framework and frequently interrupt the integration of the ODE subsystem. In this paper we develop an efficient implementation approach for the hybrid method coupled with traditional ODE solvers. We also compare the efficiency of hybrid methods with three widely used ODE solvers RADAU5, DASSL, and DLSODAR. Numerical experiments with three biochemical models are presented. A detailed discussion is presented for the performances of three ODE solvers.

DEPARTMENTS OF COMPUTER SCIENCE, VIRGINIA TECH, BLACKSBURG, VA 24060

E-mail address: wangshuo@cs.vt.edu

DEPARTMENTS OF COMPUTER SCIENCE, VIRGINIA TECH, BLACKSBURG, VA 24060

E-mail address: ypu@cs.vt.edu

DEPARTMENTS OF COMPUTER SCIENCE, VIRGINIA TECH, BLACKSBURG, VA 24060

E-mail address: lwatson@cs.vt.edu

DEPARTMENTS OF COMPUTER SCIENCE, VIRGINIA TECH, BLACKSBURG, VA 24060

E-mail address: ycao@cs.vt.edu